Evaluation of recursive predicats in deductive databases

Christophe de Maindreville

INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. : (1) 39 63 55 11

Rapports de Recherche

N° 467

EVALUATION OF RECURSIVE PREDICATS IN DEDUCTIVE DATABASES

Christophe de MAINDREVILLE

Décembre 1985

EVALUATION OF RECURSIVE PREDICATS IN DEDUCTIVE DATABASES.

Christophe de Maindreville

SABRE Project, INRIA

BP 105, 78150 LE CHESNAY, France

Abstract :

This paper presents a survey of algorithms for computing recursive axioms in deductive databases. Different approachs are presented and analyzed with similar and characteristic examples. Finally further approachs are proposed.

LE TRAITEMENT DES AXIOMES RECURSIFS DANS LES BASES DE DONNEES DEDUCTIVES.

Christophe de Maindreville

Projet SABRE, INRIA

BP 105, 78150 LE CHESNAY, France

Résumé :

Cette étude présente une synthese des diverses méthodes de controle de régles d'inférences récursives dans les SGBD déductifs.
Les différentes approches sont présentées puis discutées à l'aide d'exemples similaires et représentatif de leur spécificité.
De nouveaux axes d'études sont enfin proposés.

1.Introduction.

La mise en oeuvre des règles de déductions peut etre considérées de deux manieres differentes, [Nico83].
La premiere méthode consiste à les utiliser lors de la phase de mise à jour d'informations afin de générer puis de stocker des informations déductibles.Cette méthode est appellée méthode de génération.
Remarquons que la mise en oeuvre de cette méthode en présence de règles récursives ne pose pas de problemes particuliers, [Mink83].
La seconde consiste à les mettre en oeuvre lors de la phase d'interrogation, afin de générer temporairement les informations nécessaires à l'évaluation de la requete.Cette méthode est appellée dérivation.

A la difference de la méthode précedente l'arret du processus d'inférence pose des problemes de mise en oeuvre. En effet les techniques usuelles d'inférences utilisées en Intelligence Artificielle peuvent mener à parcourir des chemins de dérivations potentiellement infinis.
Dans le contexte des bases de données déductives, si les axiomes utilisés ne comportent pas de fonctions externes, "function-free axioms", le nombre de tuples inférés ne peut etre que fini.Il est donc théoriquement possible de trouver des conditions d'arret pour tout chemin de dérivation.

Un second probleme posé par la présence d'axiomes récursifs dans les bases de données déductives est la nécessité d'introduire de nouvelles techniques d'optimisation et d'évaluation de requetes.En effet la

technique usuelle donnant la priorité aux sélections par rapport aux jointures se révele inapliquable dans un contexte récursif.

2. Les méthodes compilées.

2.1 Introduction.

Le but de ces méthodes est de transformer une requete dont l'évaluation met en jeu un ensemble d'axiomes récursifs en un programme de requetes n'invoquant que des relations explicitements stockées dans la base de données.
L'ensemble des prédicats appartenant à l'IDB (intentional database) sont donc compilés en expressions composées uniquement de prédicats appartenant à l'EDB (extensional database) et en une suite d'opérateurs de l'algebre relationnelle. Les variables appartenant à ces expressions étant instanciées par les valeurs induites de la requete, le systeme de gestion de base de données peut alors évaluer cet ensemble de requetes de maniere tout à fait classique.

Cette approche permet donc de dissocier totalement les processus de résolution et d'évaluation des requetes. Ceci autorise l'utilisation du processeur d'optimisation de questions des SGBD relationnels classiques, les acces à la base de données pouvant être differés et regroupés en un seul acces.

Ce type de méthode souleve principalement deux sortes de difficultes:

- l'obligation d'effectuer une optimisation statique des axiomes récursifs, ie, de profiter au maximum lors du processus de

résolution des instanciations procurées par la requete initiale afin de ne generer que les réponses désirées par l'utilisateur; ceci correspondant à effectuer au cours du processus d'évaluation les opérations de sélections le plus tot possible.

- la représentation de ces programmes iteratifs au sein d'un langage hote ou à l'aide d'opérateurs specifiques du type opérateur de point fixe qui viendraient enrichir l'algebre relationnelle classique ainsi que de leur structure de controle associée.

La plupart des méthodes que nous étudierons utilisent les techniques de chainage arriere ou "goal-driven deduction" qui sont rappellées brievement ici.

Soit Pi un prédicat apparaissant dans une requete R: P1&P2&...&Pi&...&Pm
Supposons que ce prédicat ne puisse être résolu par aucune relation de l'EDB.
S'il existe une clause C: Q1&Q2&...&Qn --->Qn+1 et une substitution s telle que Pi.s = Qn+1 alors les prédicats Pi et Qn+1 sont dits unifiables. On peut alors remplacer Pi par le prémisse de la clause C pour obtenir une nouvelle requete ,

R' :(P1&P2...&Pi-1&Q1&..&Qn&Pi+1&...&Pm).s

telle que chaque solution de R' soit une solution de R.
Le processus est réitéré jusqu'a ce que tous les prédicats que comporte la requete puissent etre résolus par les prédicats de l'EDB. C'est alors que s'effectue le processus d'évaluation sur les relations

explicitement stockées dans la base de données.

Cette technique de résolution peut conduire lors de l'évaluation des axiomes récursifs à parcourir des chemins de dérivation de longueur infinie. En effet le litteral non explicite étant réevalue à chaque étape de la récursion il faut pour éviter de "boucler" indéfiniment sur la meme requete que les arguments figurant dans ce litteral soient au moins partiellement instanciés afin de profiter des répercussions procurées par les prédicats appartenant à la base de faits.
Le chainage arriere est donc inutilisable tel quel pour calculer par exemple la fermeture transitive d'une relation binaire.

De meme remarquons que la génération d'une suite d'expressions ne contenant que des relations de base ne permet pas toujours de choisir comme condition d'arret l'absence de résultats nouveaux à l'étape j du processus d'évaluation.
En effet il peut arriver qu'une expression d'ordre j+i puisse produire des tuples nouveaux.
Illustrons ceci par un exemple:

Exemple 1 :
Soient la relation ANCETRE définie par les axiomes suivants :

PERE(x,y) ---> ANCETRE(x,y)

PERE(x,y)&ANCETRE(y,z) ---> ANCETRE(x,z)

ou la relation PERE est explicite, et la requete, ANCETRE(a,b);

Le developpement de la définition récursive de la relation ANCETRE en relations explicites fournit alors la suite d'expressions suivante:

PERE(a,b)

PERE(a,x)&PERE(x,b)

PERE(a,x)&PERE(x,y)&PERE(y,b)

.....

Il est alors immediat que l'absence de réponse à l'étape i ne permet pas de stopper définitivement le processus de recherche.
Pour remedier à ce genre de desagrement certains auteurs proposent d'alterner le chainage arriere avec le chainage avant, technique qui permet de "saturer" les relations virtuelles sur lesquelles porte la requete. Ceci permet donc de profiter des avantages procurés par ces deux méthodes:

-la seule génération des faits utiles pour la requete par le chainage arriere.

-l'absence de probleme de condition d'arret du processus de dérivation pour le chainage avant étant entendu que l'on profite alors de la finitude du domaine sur lequel on procede à l'inference.

2.2 Méthode de Chang.

Ces travaux [Chang81], font suite à [Chang78] ou l'auteur proposait un langage d'interrogation de base de données relationnelle, DEDUCE2,

autorisant la manipulation d'informations sur des relations déduites. La méthode ne reglait pas le probleme des vues définies récursivement.

Principe de la méthode:
Il s'agit de représenter l'ensemble forme de la requete et des axiomes impliqués en un graphe orienté, d'en déduire un ensemble de règles de transformation (rewriting rules) afin de generer une requete équivalente, compilable en un programme executable par le SGBD.

Cette méthode s'appuie sur deux concepts importants:
-le principe d'unification [Chang73].
-la théorie des langages réguliers [Kain].

Le principe d'unification permet de generer un graphe de connection [Kowa75] dont les arcs relient les literaux figurant à droite d'une clause aux litteraux unifiables figurant à gauche.

Exemple 2 :

Soient les relations:

EMPLOYES(nom = x,manager = y,departement = z).

VENTES(departement = u,objet = v).

et la relation virtuelle COMMANDE(superieur = s,nom = t) définie par:

EMPLOYES(y,x) ---> COMMANDE(x,y).

EMPLOYES(z,y)&COMMANDE(x,y) ---> COMMANDE(x,z).

La relation COMMANDE est donc définie par un axiome de type récursif.

Soit la requete:

"Trouver toute les personnes qui sont commandées par jean et qui vendent des fleurs".

Celle-ci se représentera dans la syntaxe du langage DEDUCE2 par:

EMPLOYES(*nom = x,departement = y)&

VENTES(departement=y,objet = fleur)&

COMMANDE(superieur = jean,nom = x).

L'ensemble de la requete et des axiomes la concernant se représentera par le graphe suivant:

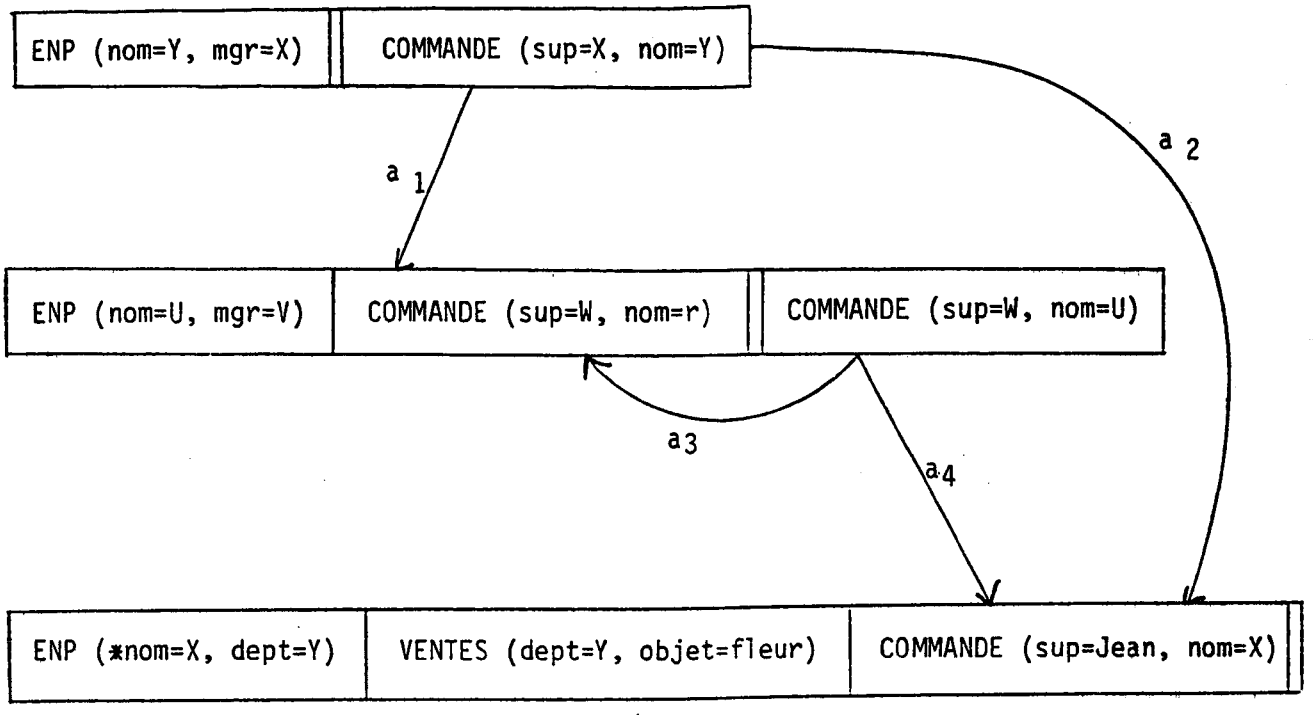


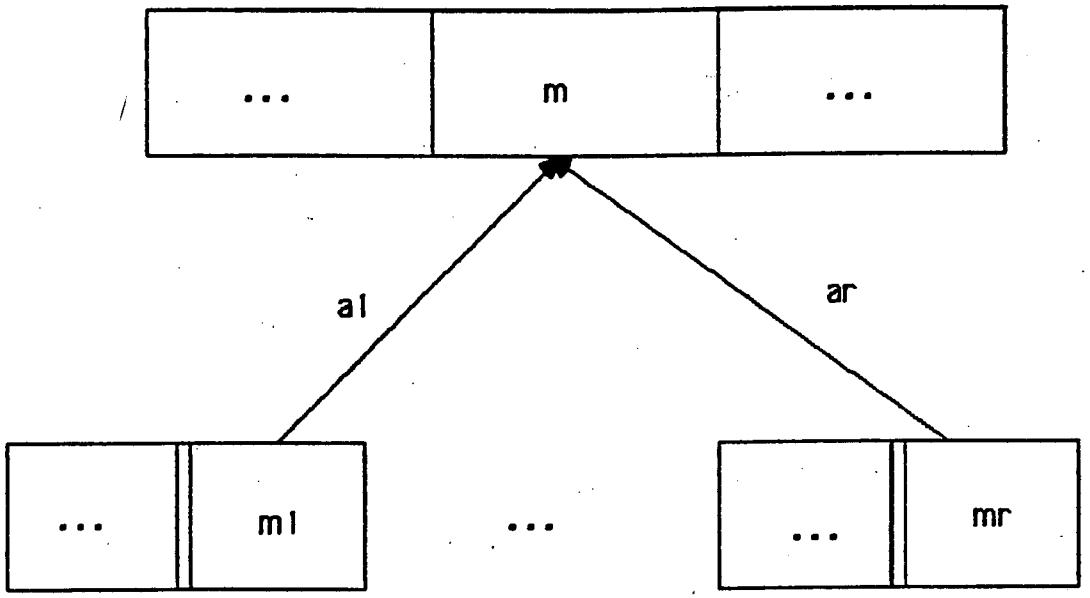
Schéma 1

Obtention des règles de transformations.

(i) pour les literaux gauches: Soit n un litteral figurant à gauche d'une clause;

si m1,...,mr sont tous les litteraux droits pointant vers n on obtient alors la règle suivante:

W(n)=a1W(m1)U...UarW(mr).



Schema 2

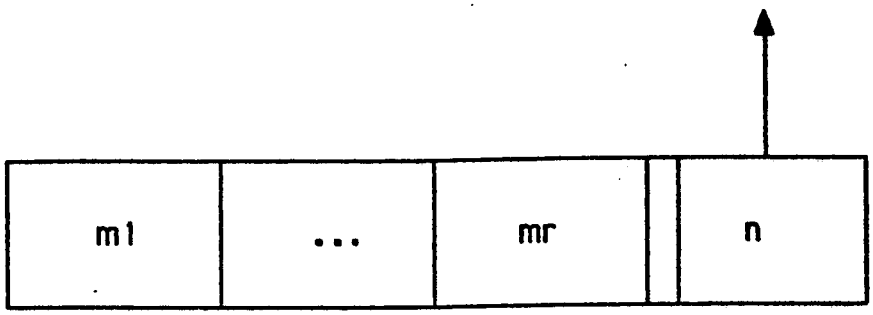
(ii) pour les literaux droits:

Soit n un literal figurant à droite d'une clause;

s'il existe un arc le quittant on obtient alors la règle :

W(n) = P1...Pr. ou les Pi sont définis par: Pi = W(mi) si mi contient une relation virtuelle

= mi sinon.



Schema 3

(iii) pour la requete:

Soit R1&...Rm une requete qui se représente par la clause suivante:

R1&...Rm --->

Schema 4

La règle obtenue est alors la suivante:

R = Q1...Qm

ou les Qi sont définis comme suit: Qi = W(Ri) si Ri contient une relation virtuelle

= Ri sinon.

On obtient ainsi une suite de règles exprimées à l'aide de symboles terminaux et non terminaux d'un alphabet ce qui permet de distinguer les litéraux explicites des litéraux implicites.

Exemple 3 :

Pour la requete précedente on obtient la suite de règles :

W(b2) = x1W(a2) U x3W(b3)

W(c3) = x2W(a2) U x4W(b3)

W(a2) = a1

W(b3) = b1W(b2)

Q = c1c2W(c3)

Par substitutions on obtient les règles simplifiées suivantes:

W(b2) = x1a1 U x3b1W(b2)

Q = c1c2x2a1 U c1c2x4W(b2)

Le probleme est maintenant de s'affranchir du terme défini récursivement ie W(b2).

Pour cela introduisont quelques définitions.

Définition 1 :

Soit un alphabet A = {a1,...an}. Les expressions régulieres sur cet alphabet sont données par:

(i) a1,...an, k,0

(ii) si P et Q sont régulieres alors P U Q , PQ,

P* = k U P U P2 U ..... , le sont aussi.

(iii) toutes les expressions régulieres sont générées par les régles (i) et (iii).

Définition 2 :

Soit q une fonction définie sur les expressions régulieres par la suite de régles suivantes:

(i) q(ai) = 0

(ii) q(0) = 0

(iii) q(k) = 1

(iv) q(P U Q) = 0 si q(P) et q(Q) = 0

= 1 sinon

(v) q(PQ) = 1 si q(P) et q(Q) = 1

= 0 sinon

(vi) q(P*) = 1

Theoreme 1 :

L'équation D = a U bD ou a et b sont des expressions régulieres et q(a) = 0 a pour solution : D = b*a .

Remarque:

Ce théoreme n'est autre qu'un théoreme de point fixe sur un alphabet et permet donc de passer d'une représentation inductive d'ensemble à une représentation explicite.

Les notions nécessaires de théorie des langages, ainsi que la démonstration du théoreme peuvent etre trouvés dans [Brzo66].

Ce théoreme appliqué à l'exemple précedent donne pour Q l'expression suivante:

Q = c1c2x2a1 U c1c2x4b1(x3b1)*x1a1

c'est à dire en remplacant les xi par les litéraux concernés:

Q = c1c2(c3a2)a1 U c1c2(c3b3)b1((b2b3)b1)*(b2a2)a1

Q s'écrit donc comme la réunion de deux expressions régulieres;

le premier terme ne posant aucune difficulté , [Chang78], considérons uniquement le deuxieme.

Par substitution et renomage des variables on obtient les expressions suivantes:

c1	EMPLOYES(*nom = s,dept = t)
c2	VENTE(dept = t,objet = fleur)
(c3)	(COMMANDE(superieur = jean,nom=s))
(b3)	(COMMANDE(superieur = w,nom = u))
b1	EMPLOYE(nom = u,manager = v)
((b2))	((COMMANDE(superieur = w,nom = v))*
((b3))	((COMMANDE(superieur = ww,nom = uu))
(b1)	(EMPLOYES(nom=uu,manager = vv)
b2	COMMANDE(superieur = ww,nom = vv)

a2 COMMANDE(superieur = x,nom = y)

a1 EMPLOYES(nom = y,manager = x)

Apres unification des termes compris entre parentheses:

EMPLOYES(*nom = s,dept = t)

VENTES(dept = t,objet = fleur)

EMPLOYES(nom = s,manager = v)

EMPLOYES(nom = v,manager = vv)*

EMPLOYES(nom = vv,manager = jean)

Cette suite de commandes conduit alors naturellement au programme suivant:

```
EVALUATE EMPLOYES(*nom = vv,manager = jean);
*V <--- *VV
DO WHILE *V = empty
   EVALUATE (EMPLOYES(nom = s,manager = v)&
             EMPLOYES(*nom = s,departement = t)&
             VENTES(departement = t,objet = fleur)) for each v E *V
   PRINT *S
   EVALUATE EMPLOYES(*nom = v,manager = vv) for each vv E *VV
   *VV <--- *V
END;
```

Remarque:

Cette méthode s'applique seulement à une classe d'axiomes récursifs, classe dite régulière, c'est à dire ne comportant au plus qu'une relation virtuelle dans leur membre de gauche.

Il est cependant possible d'étendre cette classe d'axiomes, [Mink83], à ceux qui comportent plusieurs relations non implantées dont au plus une n'est pas résolvable indépendament de cette clause.

Exemple 4 :

Soit l'ensemble de règles suivantes:

(i)

F ---> A

F&A ---> A

(ii)

P ---> N

A&N ---> N

ou les relations P et N sont des relations implantées.

Il suffit alors pour résoudre (ii) d'avoir résolu le systeme (i);par ce type de remarque on peut ainsi enrichir la classe d'axiomes récursifs que l'on peut traiter avec la méthode de Chang et ainsi traiter certains types de récursivités croisées.

2.3 Méthode de Henschen et Naqvi

Les auteurs proposent dans [Hens83] un algorithme qui à partir d'une requete exprimée sous forme de point fixe la transforme en un programme itératif d'opérations de l'algebre relationnelle.
Cet algorithme est détaillé dans [Hens84].

Rappel sur la sémantique du point fixe.

L'utilisation de la programmation logique et en particulier des clauses de Horn comme langage de définition et d'interrogation de bases de données relationnelles a conduit certains auteurs à developper une classification des requetes exprimables dans les langages d'interrogation .
Il a été démontré dans [Chan82a] et [Chan82b] que l'algebre relationnelle ne permettait pas d'exprimer des requetes du type point fixe et que la puissance d'expression d'un langage basé sur des clauses de Horn est exactement celle procurée par l'application d'un opérateur de point fixe à des formules positives du premier ordre.
De telles requetes apparaissant naturellement lors de l'interrogation d'une base de données, il est souhaitable de pouvoir enrichir l'algebre relationnelle de ce type d'opérateur [Aho79], et ainsi de préserver l'aspect non procédural du langage d'interrogation.
Une autre optique, choisie par les auteurs de cet article, est de simuler cet opérateur par un ensemble de procédures comportant une suite de commandes relationnelles classiques.

Définition 3 :

Soit une expression du type R=f(R) (1)

ou R est une relation et f la composée d'opérateurs de l'algebre relationnelle.

On appelle plus petit point fixe de l'équation (1) une relation R* qui verifie:

(i) R*=f(R*)

(ii) si P est une relation telle que P=f(P)

alors R* < P

Theoreme 2 (Tarski) :

Soit une fonction vérifiant:

R1 < R2 ===> f(R1) < f(R2)

alors cette fonction est dite monotone et l'équation

R = f(R) possede un point fixe unique.

Une démonstration de ce théoreme peut etre trouvée dans [Apt82].

Exemple 5 :

La fermeture transitive d'une relation binaire Ro est le point fixe de l'équation R = f(R) avec f(R) = P1,4(s2=3(RxRo)).

La recherche d'une relation verifiant une équation de type (1) peut se résumer en specifiant les deux étapes suivantes:

(2) initialisation Ro < R

induction f(R) < R

Remarquons alors qu'une simple boucle "while" permet d'exprimer une procedure synthetisant la recherche d'un point fixe.

```
(3)     R <--- Ro
          DO
           R' <--- R
           R  <--- R U f(R)
          WHILE (R'<>R)
```

Les travaux de Henschen et Naqvi consistent donc à partir d'un ensemble d'axiomes de type (2) à constituer un programme itératif de type (3) et ceci en profitant des optimisations possibles procurées par la requete.

Leurs propositions peuvent se résumer en deux étapes:
- la premiere consiste à utiliser un graphe de connection similaire à celui qu'utilise Chang pour réecrire un ensemble de clauses récursives sous forme de clauses ou la récursion ne porte que sur un seul prédicat.
- la seconde produit à partir d'une clause réguliere un programme de type (3).Ces programmes seront alors stockés sous forme précompilée et seront appellés et executés lors de l'évaluation d'une requete.

Comme nous le préciserons plus tard le stockage de ces programmes peut devenir problematique puisqu'a un ensemble de clauses régulieres peut etre associe plusieurs programmes correspondant à des requetes

differentes.

Developpons sur un exemple l'étape 2;

Soient les clauses suivantes désormais célèbres:

PERE(x,y) ---> ANCETRE(x,y)

ANCETRE(x,y)&PARENT(y,z) ---> ANCETRE(x,z)

et la requete ANCETRE(x,a)

Les techniques de chainage arriere conduisent alors à l'algorithme suivant:

1. -retrouver PERE(w,a) et mettre les tuples résultats dans la relation ANCETRE
2. -"empiler" la constante 'a'
3. -"depiler" une valeur de la pile et en affecter la variable z
4. -retrouver PARENT(y,z) et utiliser les valeurs de 'y' pour retrouver PERE(w,y), depiler les valeurs de y;
   inserer les tuples résultats dans la relation ANCETRE
5. -aller en 3

Considerons maintenant la requete ANCETRE(a,y).

Il est alors clair que l'algorithme précedent ne s'applique pas à cette requete:

la requete ne provoque en effet aucune instanciation dans la relation PARENT et ne permet donc pas de poursuivre le processus de résolution tel qu'il a été défini ci dessus, (cf les limitations des techniques de

chainage arriere).Il est alors necessaire d'instancier le prédicat ANCETRE figurant dans la deuxieme clause à l'aide des résultats procurés par la résolution de la premiere clause afin de pouvoir résoudre le prédicat PARENT.
L'algorithme doit etre donc légèrement modifié.

De maniere générale, considerons des clauses régulieres du type suivant:

(i) F1&F2&...&Fn ---> R

(ii) A1&A2&...&R&B1&...&Bn ---> R

ou l'on appellera F1&F2&...&Fn la partie déterminée de l'ensemble de clauses et A1&A2&...&B1&B2&...Bn la partie induite.
En se reportant à l'exemple précedent on remarque alors que les instanciations procurées par la requete influent sur l'ordre dans lequel doivent etre utilisées les parties déterminées et induites.Cet ordre peut etre défini comme suit:
si le sous-but A1&...&R&...&Bn est résolvable, ie comporte au moins une variable instanciée, alors l'ordre est : évaluer PI, puis PD
sinon évaluer PD puis PI.

Soit donc l'algorithme général: Algorithme 3

```
R = R U eval(PD)
inserer-dans-pile P (constantes issues de la requete)
DO WHILE P<>0
   Z <--- contenu-de-la-pile P
   Y = eval(PD ou PI,Z)
   R = R U eval(PI ou PD,Y)
   inserer-dans-pile P (nouvelles constantes)
OD
```

Remarques:

Cette approche autorise, à la difference de celle de Chang, la compilation d'axiomes comportant plusieurs prédicats récursifs, reductibles à des axiomes réguliers.

Le controle de ces récursivités multiples présente de notables difficultés et conduit à des algorithmes particulierement complexes, les auteurs en laissant d'ailleurs la résolution aux chercheurs "more concerned in operating systems". Il est à noter que pour les récursions portant sur une seule variable les deux méthodes sont identiques.

Les complexités relatives des deux méthodes ont êtés étudiées dans [Viell84].

La méthode de Henschen et Naqvi s'avere etre plus performante en présence de récursivité portant sur plusieurs variables, les conditions d'arret portant généralement sur des ensembles de taille plus réduite.

Il apparait également que le degré de connectivité des graphes représentant les relations explicites est un facteur influencant fortement les performances relatives de ces deux méthodes.

2.4 Méthode d'Alexandre.

L'originalité de cette méthode consiste à utiliser conjointement les techniques de chainage arriere et de chainage avant. L'idée directrice étant de déterminer en chainage arriere les règles pertinentes qui en chainage avant répondront à la requete.

La solution proposée par les auteurs ,[Bull85] ,peut se decomposer en les étapes suivantes:

1. La requete est considerée comme un but à évaluer en chainage arriere.
2. Cette requete est traduite en un ensemble de clauses qui seront executées en chainage avant.
3. Cet ensemble de clauses est traduit en une suite d'opérations de l'algebre relationnelle.

Les auteurs dans cet article ne s'interessent qu'au passage de l'étape 1 à l'étape 2, le passage de l'étape 2 à l'étape 3 ayant été étudié puis réalisé dans le cadre d'un moteur d'inference appelle BOUM, [Jamie83].

Nous nous proposons de traiter un exemple désormais classique ce qui permettra d'illustrer la spécificité de cette méthode.
Soient donc les clauses suivantes:

PERE(x,y) ---> ANCETRE(x,y)

PERE(x,y)&ANCETRE(y,z) ---> ANCETRE(x,z)

La requete étant par exemple: ANCETRE(a,x).

Ce but est alors coupe en deux, un probleme et un ensemble de solutions, ce qui permettra de simuler le chainage arriere par les techniques de chainage avant.
Le probleme s'écrit sous la forme d'un prédicat de la forme Pb-Ancetre(a) et une solution est un fait du type Sol-Ancetre(a,b).

A l'aide de ces deux prédicats il est possible d'écrire les règles qui permettront de générer les problemes et les solutions en chainage avant:

Les deux axiomes initiaux sont alors transformés en:

1 Pb-Ancetre(x)&PERE(x,y) ---> Sol-Ancetre(x,y)
ce qui peut s'interpreter par:si l'on a le probleme de trouver les descendants de x et que x est le pere de y alors y est une solution.

2 Pb-Ancetre(x)&PERE(x,y)&ANCETRE(y,z) ---> Sol-Ancetre(x,z)
ce qui signifie:si l'on a le probleme de trouver les descendants de x , que x est le pere de y et que y est l'ancetre de z, alors z est une solution.

Cette derniere règle comportant elle meme un but, ce dernier subit la meme transformation que précedement ce qui conduit à réecrire la règle 2 avec les deux règles suivantes:

2.1 Pb-Ancetre(x)&PERE(x,y) ---> Pb-Ancetre(y)

2.2 Sol-Ancetre(y,z) ---> Sol-Ancetre(x,z)

On remarque alors que la règle 2.2 est incorrecte, la variable x ne figurant pas dans le prémisse de la clause.Il faut pouvoir specifier que cette variable provient de la clause 2.1.A cet effet il est introduit un prédicat supplémentaire, dit de contexte, qui préserve cette information.

Les deux règles s'écrivent alors:

2.1' Pb-Ancetre(x)&PERE(x,y) ---> Pb-Ancetre(y)&Cont(y,x)

2.2' Sol-Ancetre(y,z)&Cont(y,x) ---> Sol.Ancetre(x,z)

En rajoutant la règle 1 on obtient alors un ensemble de règles qui en chainage avant répondront à la requete posée.

Remarques:

Cette méthode permet la compilation d'axiomes non réguliers, ce qui autorise par exemple la définition de la relation ANCETRE par des axiomes du type

ANCETRE&ANCETRE ---> ANCETRE.

Il reste cependant à gerer les processus de transformation ainsi que les prédicats de contexte pour des récursivités multiples ce qui devrait alourdir sensiblement les techniques à mettre en oeuvre.

De plus rappellons que les règles de productions obtenues par cette méthode doivent etres traduites en algebre relationnelle avant de pouvoir etre mises en oeuvre sur un SGBD.

Une solution compilée aux problemes des axiomes récursifs a été également proposée par Marque-Pucheu ,[Marq83] .Une formalisation du problême est proposée et des solutions partielles sont présentées.
Il s'agit d'une méthode purement algébrique basée sur une extention de la notion d'opérateurs relationnels et sur les automates d'états finis. Il est montré que pour une suite d'opérations relationnelles donnée (représentée par un arbre ou un ensemble d'arbres) il existe toujours un arbre optimisé équivalent.

De manière pratique ,il est nécessaire de recenser un ensemble de règles d'optimisation afin de développer des heuristiques de compilation.

3. Les méthodes interpretées.

3.1 Introduction.

Dans cette approche, les processus de résolution et de recheche des informations dans la base de faits sont intimement lies.La réponse à une requete mettra en jeu de facon simultanée les axiomes et les faits impliques.

Cette méthode provenant des techniques utilisées en Intelligence Artificielle est similaire à celle suivie dans les "demonstrateurs de theoremes" et est également celle adoptée par le langage PROLOG.

Le probleme initial est décomposé par chainage arriere en sous problemes qui seront résolus par des acces successifs à la base de données et dont la reunion des réponses fournies formera la réponse complete à la requete.

Afin de pouvoir utiliser cette méthode dans le contexte des bases de données déductives il a été propose deux types d'optimisations :

- -utilisation d'une fonction de sélection qui indique lors de la résolution d'une clause conjonctive l'ordre dans lequel ils seront evalués [Mink78a,Mink78b].
- -regroupement des acces concernant une sous-requete afin de pouvoir obtenir les solutions de maniere ensembliste [Mink75,Mink78b].

Notons que les problemes attachés au traitement des axiomes récursifs persistent pour l'approche interpretée, ces problemes étant liés au mecanisme d'inférence utilisé, le chainage arriere, et non à l'ordonnancement des processus informatiques mis en oeuvre.

3.1 Méthode de Lozinskii.

Cette méthode, [Lozin84] ,[Lozin85], consiste à partir de la requete initiale à détérminer de maniere dynamique les sous-problemes engendrés et les faits necessaires à la résolution de ces sous-problemes.
Ces derniers seront résolus "at run time" et leur reunion formera l'ensemble des solutions demandées.Le probleme consiste donc pour chaque sous requete à determiner ce qui est appelle "faits relevants", puis s'ils existent, d'engager en chainage avant le processus de recherche des informations élementaires dans la base de données.

Cette approche combine donc la technique de chainage arriere pour décomposer la requete en sous-questions et pour determiner les faits relevants et le chainage avant pour évaluer ces sous-questions sur la base de données.
Ceci permet donc de s'affranchir théoriquement des problemes de conditions d'arret, et autorise également l'arret anticipé du processus engagé en cas de non existence de faits relevants à la requete posée.

De maniere classique, les axiomes relevants de la requete sont representés par un graphe ET/OU, ou les ET-noeuds reprêsenteront les relations mises en jeu et les OU-noeuds les axiomes proprement dit.

Dans le cas ou une relation est définie de maniere récursive, l'occurence de celle-ci apparaissant dans le prémisse d'une clause sera représentée par un noeud principal et celle apparaissant en partie droite par un noeud secondaire.

Exemple 5 :

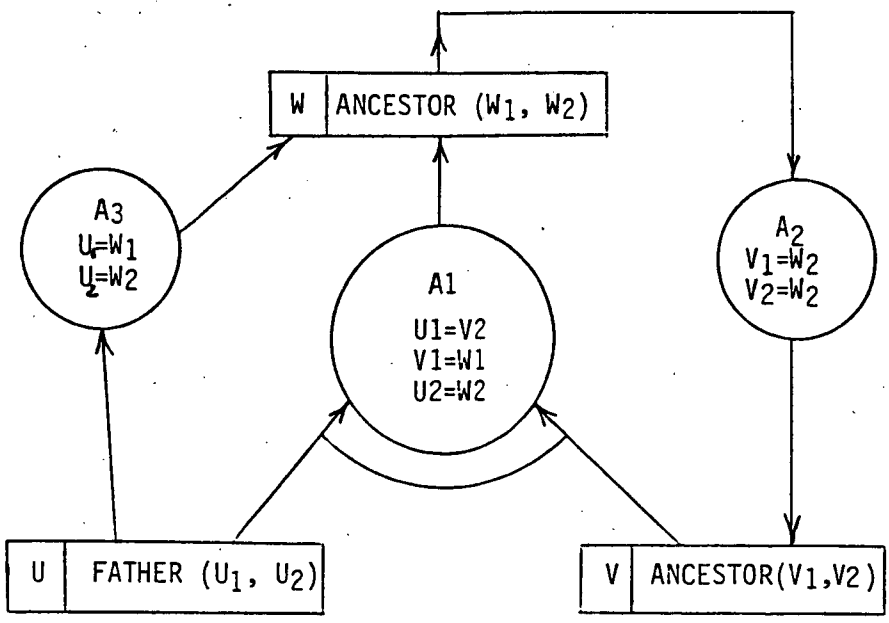
la relation ANCETRE pourra etre définie par les axiomes suivants:

PERE(u1/nom,u2/nom)&(u1=w1)&(u2=w2) ---> ANCETRE(w1/nom,w2/nom)

ANCETRE(x1/nom,x2/nom)&PERE(y1/nom,y2/nom)&(x2=y1)&(x1=z1)&(y2=z2)

---> ANCETRE(z1/nom,z2/nom)

et cette définition conduira au graphe suivant:



Schema 5

Recherche des faits relevants.

Définition 4 :

Assurons que toutes les variables apparaissant dans les clauses relevantes d'une requete portent des noms differents.

Soit alors l'ensemble MIG(x) défini comme suit:

(i) S'il existe un axiome,

Q(...)&...&P(...,x,...)&...&(x=y)&... ---> R(...,y,...)

alors y E MIG(x).

(ii) Si y E MIG(x) et z E MIG(y) alors z E MIG(x).

Cet ensemble représente donc les differents termes vers lesquels x peut "migrer" lors du processus de résolution.

Remarquons que cet ensemble peut etre aisement calculé en parcourant un graphe du type de celui qui a été défini précedement. Une procedure est d'ailleurs proposée à cet effet dans l'article étudié.

Définition 5 :

Soient une requete R:R(x1,x2,....,xi=ai,...) et un fait P(y1=b1,...,yj=bj,...).

P est dit relevant de Q si P contient yj tel que xi E MIG(yj) et bj = ai.

Exemple 6 :

Pour la relation ANCETRE on a:

MIG(x1) = êx1,y1,z1} et MIG(x2) = êx2,y2,z2}

Soient les requetes Q1:ANCETRE(x,a) et ANCETRE(a,y).

Les faits relevants de Q1 sont les faits du type: PERE(z,a)

et ceux relevants de Q2 : PERE(a,z).

Nous pouvons alors énoncer l'algorithme general propose par Lozinskii:

Soient Q la requete considerée, F l'ensemble des faits appartennant à l'EDB.

Algorithme 3 :

1. Selectionner les faits relevants FR de la requete Q.
2. S'il existe un axiome executable sur FR
   alors executer et rajouter les faits nouveaux dans FR.
   sinon rechercher dans F les faits necessaires à l'execution
   et rajouter les faits nouveaux dans FR.
3. Aller en 2 tant que des faits nouveaux sont produits sinon FIN.

Exemple 7 :

Soient les relations M,P,R et les relations virtuelles N et S définies par les axiomes suivants:

A1:S(s1,s2)&P(p1,p2)&(s2=p1)&(s1=n1)&(p2=n2) ---> N(n1,n2)

A2:M(m1,m2)&N(n1,n2)&(m2=n1)&(m1=s1)&(n2=s2) ---> S(s1,s2)

A3:R(r1,r2)&(r1=n1)&(r2=n2) ---> N(n1,n2)

Soient par exemple les extensions suivantes des relations M,P,R:

M		P		R	
m1	m2	p1	p2	r1	r2
b	c	d	a	e	d
c	e	g	h	t	g
f	t				

Soit la requete Q:S(x,a).

Appliquons l'algorithme précedent à cet exemple.

1. P(d,a) est le seul fait relevant de Q.
2. Pour executer A1 il faut un fait du type S(x1,d) pour effectuer la jointure.Pour obtenir un tel fait résolvons Q2:S(x1,d). R(e,d) est relevant de Q2 (car s2 E MIG(r2)). L'execution de A3 sur R(e,d) fournit N(e,d). Pour executer A2 sur N(e,d) il faut un fait du type M(x2,e). Résolvons donc Q3:M(x2,e), ce qui fournit pour solution M(c,e). A2 genere S(c,d) et A1 N(c,a).Il faut donc un fait M(x3,c) ce qui conduit à la sous requete Q4:M(x3,c) qui donne pour solution M(b,c). Finallement A2 genere l'unique réponse S(b,a).
3. FIN.

Remarques:

Cet algorithme est naturellement develloppé dans l'article étudié et

des preuves de sa complètude et de sa terminaison sont proposées.Il est donne diverses optimisations en vue d'une implémentation possible afin par exemple de générer les réponses de maniere ensembliste et non "tuples à tuples".
Il est également propose des techniques d'implémentations parallelles des processus d'inference.

Notons que cette méthode s'applique à toute classe d'axiomes récursifs. Cette solution est à rapprocher de celle proposée dans [Bull85] et peut etre considérée comme une version intérprétée de cette dérniere.

Une autre solution intérprétée est proposée par McKay et Shapiro ,[McKay81] .
Le formalisme utilisé repose sur les "active connection graph" dans lesquels les arcs relient les prédicats unifiables.Cette méthode n'effectue aucune distinction entre l'EDB et l'IDB et semble donc etre plus particulierement déstinée aux applications de l'intelligence artificielle.
Elle n'effectue aucune restriction sur les axiomes ,mais ne permet pas de donner la priorité aux opérations de selections :il est par exemple necessaire de calculer la totalité de la relation Ancetre afin de répondre a une requete du type Ancetre(toto,y).

Il reste que les téchniques proposées de part leur aspect interpreté ne semblent pas pouvoir bénéficier pleinement des pérformances propposées dans le domaine de l'optimisation de requetes par les SGBD relationnels.

4.Autres méthodes.

4.1 Méthodes de Reiter et de Minker ,Nicolas.

Les deux méthodes qui sont étudiées dans ce paragraphe permettent de s'affranchir des problemes liés aux axiomes récursifs en séparant de facon "adequate" l'EDB de l'IDB.

Il s'agit de rendre explicites les relations définies par des axiomes récursifs pouvant mener à des chemins de dérivations potentiellement infinis.

Soit l'ensemble de clauses de Horn suivant:

P1&C1 ---> P'2

P2&C2 ---> P'3

P3&C3 ---> P'4

. .

. .

. .

Pn&Cn ---> P'1

ou pour chaque i Pi et P'i sont des literaux unifiables de l'EDB. Un tel systeme initialisé avec l'un des Pi conduit à parcourir des chemins de longueur infini.

L'idée est alors de couper ce chemin de dérivation en rendant l'une des régles improductive lors du processus de dérivation.

La méthode de Reiter [Reit78b], consiste à rendre l'un des Li

"extensionnellement complet" vis à vis d'une des règles Ri ou il figure, ie, tous les faits dérivables par Ri figurent deja dans la relation Li. La règle Ri est alors considerée comme contrainte d'integrité pour assurer cette propriété lors des mises à jours sur la base de données, [Nico78].

Minker et Nicolas [Mink83], proposent d'utiliser cette règle en génération.

Ces deux méthodes rendent donc explicites des faits définis implicitement, soit de maniere statique en utilisant l'intégrité, soit de maniere dynamique en alternant génération et dérivation.

Ces solutions s'appliquent à toutes les classes d'axiomes récursifs mais alterent serieusement l'interet de définir des faits de maniere implicite.

Remarque :

Dans la méthode proposée par Minker et Nicolas ,une relation pouvant etre définie conjointement par des règles de dérivation et des règles de génération ,il est nécessaire afin d'éviter toute perte d'information ,d'activer ces dernieres lors de "mises à jours virtuelles" effectuées par les règles de dérivation,[Mink83].

4.2 Méthodes de reconnaissance syntaxique.

Le but de ces méthodes ,[Mink83] ,[Ioan85] ,est de déterminer "at compile time " une condition de terminaison du processus d'évaluation qui soit indépendante du contenu de la base de donnée.
Cette condition est déterminée par une étude syntaxique des prédicats récursifs et plus particulierement de la position des variables qui y figurent.
Il est ainsi exhibé des classes de prédicats définis récursivement sur lesquels on peut "à priori" fixer une borne supérieure à la profondeur du graphe de résolution.

Ces méthodes sont donc partielles et sont liées à un mécanisme de résolution en chainage arriere.
Elles nécessitent de plus une étude syntaxique qui peut s'avérer couteuse.
En résumé ,elles ne semblent pas etre adaptées à un contexte de bases de données déductives.

5.Conclusion.

Les diverses solutions étudiées sont pour la pluspart des améliorations des techniques utilisées en Intelligence Artificielle ;les algorithmes proposés transforment une requete portant sur une relation définie récursivement en suite de requetes plongées dans un langage procédural de programmation (méthode de compilation) ,ou en une suite d'appels au SGBD (méthode inter

Le langage de manipulation est du type PROLOG et ne permet donc pas aisement d'inclure des fonctions spécifiques aux bases de données du type aggregats;de plus les méthodes d'inférences utilisées sont basées sur le principe de résolution de Robinson.
Ces méthodes sont donc particulierement adaptées au couplage entre un moteur d'inference et un SGBD classique.

Une autre approche possible serait d'enrichir l'algebre relationnelle d'opérateurs de point fixe ,afin de transformer les clauses récursives directement en une suite d'opérations relationnelles.
Ceci permettrait une meilleure optimisation globale du processus d'inférence ainsi que la possibilité de développer des algorithmes performants de "jointures répétitives" liées à cette opération ,[Vald85].
Dans cette optique ,il est alors nécessaire de développer des techniques locales d'optimisation de requetes récursives ,telles que par exemple la priorité à donner aux sélections par rapport aux jointures dans un opérateur de point fixe.

En effet ,s'il est aisé ,avec l'approche intérprétée , de tirer parti des constantes générées par la requete ,aucune solution satisfaisante n'a encore étée proposée dans cette optique.

Il reste également à définir une mesure de complexité pérmettant de comparer efficacement les diverses méthodes publiées.On peut trouver dans [Banc85] une ébauche de solution à ce probleme.

BIBLIOGRAPHIE :

Apt82 — K.R.Apt ,M.H.Van Emden.
Contributions to the Theory of Logic Programming.
J.ACM ,Vol 29,No 3 ,Juillet 82.

Aho79 — A.V.Aho,J.D.Ullman.
Universality of Data Retrieval Languages
Conf on POPL San-Antonio,Texas,79.

Banc85 — F.Bancilhon.
Naive evaluation of recursively defined predicates.
Proc of Islamadora Workshop on Database and AI. Mars 85.

Bull85 — Une solution pour traiter les axiomes recursifs dans les bases de donnees deductives.
Rapport de recherche DRAL/IA/45.01.

Brzo66 — J.A.Brzozowski.
Class Notes of Switching and Automata Theorie ,Dept. of Electrical Engineering and Computation Science .
University of California ,Berkeley.

Chand82a — A.K.Chandra,D.Harel.
Horn Clauses and the Fixpoint Query Hierarchy.
Proc ACM Symp 82.

Chand82b — A.K.Chandra,D.Harel.
Structure and Complexity of Relational Queries.
Journal of Computer and System Sciences 25 1982.

Chang78
C.Chang.
DEDUCE 2 :further investigations of deduction in relational data bases.
In Logic and Data Bases.
Edited by H.Gallaire and J.Minker
Plenum NewYork.

Chang81
C.Chang.
On evaluation of queries containing derived relations in a relationnal data base.
In Advances in Data Base Theory
Edited by H.Gallaire,H.Minker and JM.Nicolas
Vol 1 Plenum Press , New York.

Chang73
Chang and Lee
Symbolic Logic and Mechanical Theorem Proving
Academic Press , New York.

Hens83
L.J.Henschen,S.A.Naqvi.
Synthesizing Least Fixed Point Queries into Non-recursive Iterative Programs.
Proc of 8th IJCAI 83.

Hens84
L.J.Henschen,S.A.Naqvi.
On Compiling Queries in Recursive First-Order Databases.
JACM Vol 31,No 1, Jan 84.

Ioan85
Y.E Ioannidis .
A time bound on the materialization of some recursively defined views.
Proc of VLDB 219-225. Aout 1985.

Jamie83
M.Jamier,J.M.Pugin.
Le moteur d'inference BOUM.
Rapport de DEA.
Ecole Centrale de Paris ,juin 83.

Kain
R.Kain.
Automata Theory : Machines and Languages
Mc Graw-Hill computer science series.

Kowal75 R.Kowalski.
A proof procedure using connection graph.
JACM 22,4 Oct 75.

Lozin84 E.L.Lozinskii.
Inference by generating and structuring
of deductive databases.
Technical Report 84-11 ,Hebrew university
Department of computer science ,juin 84 .

Lozin85 E.L.Lozinskii.
Evaluating queries in Deductive Databases
by Generating.
IJCAI , 173-177. Los Angeles ,Aout 85.

Marq83 G.Marque Pucheu.
Algebraic structure of answers in a
recursive logic database.
Rapport Ecole normale superieure.

McKay81 D.McKay,S.Shapiro.
Using connection graphs for reasoning with
recursive rules.
Proc 7th IJCAI,1981.

Mink75 J.Minker.
Performing inferences over relational
data bases.
ACM SIGMOD. San Jose ,California ,mai 75.

Mink78b J.Minker.
An experimental relational database system
based on logic.
Logic and Data bases,H.Gallaire,J.Minker
Plenum Press,New-York 1978.

Mink78a J.Minker.
Search Strategy and selection function
for an inferential relational system.
ACM Trans.Database Syst 3,1 .Mar 78.

Mink83 J.Minker,J.M.Nicolas.
On recursive axioms in deductive data bases
Information Systems 8.

Nico78 J.M.Nicolas , K.Yazdanian.
Integrity checking in deductive databases.
Logic and Data bases, H.Gallaire,J.Minker
Plenum Press ,New-York 1978.

Nico83 J.M.Nicolas , K.Yazdanian.
An Outline of BDGEN:A Deductive DBMS.
IFIP 83.

Reit78a R.Reiter.
Deductive question answering on relational databases.
In Logic and Data Bases.H Gallaire,J.Minker.
Ed Plenum,New York.

Reit78b R.Reiter.
On structuring a first-order database.
Proc of the 2nd Canadian Society for Computer Science
National Conference,Canada July.

Ullm85 J.D.Ullman.
Implementation of Logical Query Languages for Databases.
ACM SIGMOD 1985.

Vieil84 L.Vieille.
La compilation d'axiomes recursifs.
Rapport de stage de DEA.
Universite Paris VI ,oct 84.

Vald85 P.Valduriez.
Optimisation des operateurs relationnels dans une machine base de donnees.
These d'Etat.
Universite Paris 6 ,1985.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique